Ordonnancement temps réel des tâches sous contraintes d'énergie et de qualité de service

Akli ABBAS et Hamid HENTOUS
Unité d'Enseignement et de Recherche en Informatique,
Ecole Militaire Polytechnique, BP 17, Bordj El Bahri Alger, Algérie.
E-mail: a_abbas@esi.dz, hentoush@yahoo.fr

Résumé—Réduire la consommation d'énergie des systèmes embarqués est devenue la principale priorité des concepteurs de ces systèmes. Dans cet article nous présenterons deux algorithmes d'adaptation dynamique de la tension du processeur dans un système temps réel. Dans notre approche, nous commencerons par l'intégration des contraintes de qualité de service (QoS) dans l'ordonnancement temps réel de tâches périodiques fermes et de tâches apériodiques sous contraintes de précédences et du partage de ressources. Ensuite, nous étudierons le problème d'adaptation dynamique de la tension du processeur dans le même contexte. Les simulations étudiées montrent que notre approche permet des réductions de consommation d'énergie avec la garantie du respect des différentes contraintes.

Mots clés— Système embarqué ; ordonnancement temps réel ; tâches périodiques fermes; tâches apériodiques ; contrainte de précédences, partage de ressources, minimisation de l'énergie.

I. INTRODUCTION

Les systèmes embarqués sont caractérisés par leurs autonomies de fonctionnement, dont l'alimentation est assurée par des batteries. De ce fait, la réduction de l'énergie consommée est devenue une métrique d'optimisation cruciale dans la conception et la réalisation de tels systèmes. L'objectif visé dans ce cas consiste non seulement à déterminer l'ordre d'exécution des tâches sous différentes contraintes, mais également à fixer la fréquence de fonctionnement ainsi que la tension d'alimentation. Ces systèmes nécessitent souvent des tâches périodiques, dont certaines sont de nature critiques et d'autres peuvent manquer leurs limites temporelles. Dans ce cas une tolérance aux fautes leurs est associées. L'utilisation de ces deux types de tâches émergeantes principalement dans les domaines du multimédia et du contrôle automatique, avec des contraintes temps-réel fermes. En d'autres termes, les violations d'échéances autorisées entraînent uniquement des dégradations de Qualité de Service (QoS) sans compromettre le bon fonctionnement du système et sans mettre en danger son intégrité. Ces mêmes systèmes nécessitent des tâches apériodiques qui sont de nature non critique. Toutefois on devra diminuer leurs temps de réponse, sans pour autant compromettre l'exécution des tâches périodiques. En plus, de tels systèmes nécessitent généralement des tâches qui collaborent. Cette dernière se fait par l'échange de données et/ou par le partage de ressources. Cependant, ceci engendre des contraintes de précédence et de partage de ressources critiques, lesquelles, on doit les prendre en considération dans l'analyse de l'ordonnancement des tâches.

Cet article est organisé comme suit. Dans la section 2, nous donnerons un état de l'art sommaire et nous présenterons le modèle du système et la variation de la tension du processeur. Notre contribution concernant l'ordonnancement des tâches fermes hybrides synchrones sous QoS sera discutée dans la section 3. Dans la section 4 nous donnerons deux algorithmes d'adaptation dynamique de la tension. Les résultats expérimentaux seront discutés dans la section 5. Nous terminerons cet article par une conclusion en section 6.

II. PRELIMINARIES

Dans section, nous donnerons un état de l'art sommaire. Ensuite, nous décrivons le modèle du système et la variation de la vitesse du processeur.

A. Etat de l'art

Dans la littérature, de nombreux algorithmes ont été proposés afin de traiter le problème de l'ordonnancement temps réel des tâches. Par exemple, Liu et al. [3] ont proposés deux algorithmes RM et EDF pour l'ordonnancement des tâches périodiques indépendantes, et dans [8] les auteurs proposent une démarche pour le traitement des contraintes de précédences. Dans le cas où certaines tâches périodiques peuvent manquer leurs échéances sans pour autant compromettre le bon fonctionnement du système, on distingue les travaux de Hamdaoui et al. dans [10] qui ont proposé le modèle (m,k)-firm. Au même titre que les contraintes de précédences, l'utilisation de ressources engendre des difficultés liées à la protection des accès aux ressources par des sections critiques. Pour pallier à ces problèmes, des protocoles de gestion des ressources critiques ont été mis en œuvre. On cite par exemple le protocole d'allocation de la pile noté SRP proposé par Baker [15]. Pour l'ordonnancement des tâches apériodiques, la solution du serveur de tâches est utilisée. Cette solution peut être un serveur à largeur de bande maximale noté TBS [11]. Cette dernière solution est combinée avec le protocole SRP dans [9,6] pour l'ordonnancement des tâches hybrides (périodiques et apériodiques) partageant des ressources critiques. L'adaptation dynamique de la tension du processeur connue sous le terme de DVS (Dynamic Voltage Scaling) est l'une des stratégies de réduction de la consommation d'énergie des processeurs. Cette stratégie permet des réductions importantes de consommation [12]. Suivant cette stratégie, des techniques sont développées pour les tâches périodiques et apériodiques. Dans la littérature, on distingue les travaux de Yao et al. [5] qui proposent un algorithme qui calcule les fréquences de fonctionnement optimales du point de vue énergie pour un ensemble de tâches

périodiques. Aydin et al. [7] ont proposé une approche basée sur la condition nécessaire et suffisante proposé dans [3]. Quant à Shin et al. [4] proposent un algorithme de calcul de vitesse pour un système constitué de tâches périodiques et apériodiques. Lorsque les tâches périodiques partagent des ressources critiques, Jejurikar et al. [14] ont proposé un algorithme pour calculer le facteur de vitesse de processeur. Les autres ont utilisé la condition analytique d'ordonnancement proposée dans [2]. Dans un récent travail [1], nous avons traité le problème d'ordonnancement temps réel des tâches hybrides synchrones sous contrainte d'énergie, lequel nous avons approfondi dans [2]. Dans ce présent article, nous viserons à intégrer les contraintes de qualité de service à nos travaux ainsi cités.

B. Modele du système

Le système temps réel considéré dans cet article est un système monoprocesseur composé d'un ensemble de *n* tâches périodiques fermes noté $T=\{t_1 \ldots, t_n\}$ et par un ensemble de *v* tâches apériodiques noté $T_a= \{t_{a1} \ldots, t_{av}\}$.

1) Modèle de tâches périodiques fermes

Les tâches périodiques à contraintes stricte sont modélisées par le 6-tuple $t_i=\{r_{0,i}, C_i, D_i, P_i, m_i, k_i\}$ telle que $r_{0,i}$ est la date de réveille de la première instance de la tâche t_i, C_i est sa pire durée d'exécution noté WCET, D_i est son échéance relative et P_i est sa période avec $D_i \leq P_i$. m_i est le nombre d'échéances que la tâche t_i doit respecter et k_i est le nombre d'activations consécutives. Chaque activation d'une tâche est appelée instance, la j^{eme} instance d'une tâche t_i est notée $t_{i,j}$. Ce modèle offre un cadre général pour inclure à la fois le temps réel dur ou stricte (quand $m = k$) et le temps réel souple ou ferme (quand $m < k$). Le facteur d'utilisation du processeur par l'ensemble des tâches périodiques est

$$U = \sum_{i=1}^{n} m_i C_i / k_i P_i \leq 1 \qquad (1)$$

Ce facteur représente la condition nécessaire de faisabilité de l'ordonnancement des tâches périodiques [10].

2) Modèle de tâches apériodiques

Les tâches apériodiques sont modélisées par le1-tuple $t_{ai}=\{C_{ai}\}$, telle que C_{ai} est la sa durée d'exécution. Une tâche apériodique est spécifiée par un taux moyen d'arrivé noté λ et le taux moyen de service noté μ.

Nous utilisons dans la suite le concept « *tâches hybrides* » pour désigner les tâches périodiques et apériodiques.

3) Moèlisation des contraintes de qualité de service

Selon le modèle (*m,k*)-firm [10], chaque tâche périodique doit respecter au moins *m* échéances sur une fenêtre de *k* activations (instances) consécutives. Une faute se produit, se qui implique la violation de la contrainte de QoS et le non ordonnancement de système si pour *k* instances consécutives d'une tâche, plus de (*k-m*) instances dépassent leurs échéances. Considérant le fait qu'une des instances de la tâche ne pourra pas respectée son échéance, celle-ci est rejetée par le système. Cette politique permet ainsi de réduire la charge du système. Dans ce cas une métrique noté T_{qos} est associée à la contrainte de QoS qui représente le nombre d'instances d'une tâches qui satisfait leurs échéances (non en retard) par rapport en nombre de requêtes.

$$T_{qos} = nombre\ d'échéances\ satisfait es \,/\, nombre\ de\ requêtes \qquad (2)$$

Ramanathan propose une approche dans [13] permettant de déterminer les instances d'une tâche ferme qui sont optionnelles ou critiques. Ainsi, une instance $t_{i,j}$ (c.à.d. la j^{eme} instance de la tâche *i*) est considérée critique si :

$$j = \left\lfloor \left\lceil j * \frac{m_i}{k_i} \right\rceil * \frac{k_i}{m_i} \right\rfloor, \; j = 0,1,2,\ldots \qquad (3)$$

Dans le cas contraire, l'instance est considérée optionnelle et elle peut être rejetée par le système. On remarque que la technique de marquage des instances optionnelles et critiques ne dépend que du rapport m_i/k_i, mais pas de C_i et T_i.

Nous considérons que les tâches qui sont de nature fermes (avec $m<k$) ne sont pas liées par des contraintes de synchronisation.

4) Modèlisation contraintes de partage des ressources et de précédences

Le système possède un ensemble de ressources partagées par les tâches périodiques strictes et les tâches apériodiques. Ces ressources sont utilisées en exclusion mutuelle par l'utilisation des sémaphores. Lorsqu'une tâche accède à la ressource, elle exécute sa section critique. La k^{eme} section critique d'une tâche t_i ou t_{ai} qui utilise la ressource R_k est représentée par $\beta_{i,k}$. On considère qu'une tâche est bloquée si elle doit attendre la libération de la ressource par la tâche de plus haute priorité. On appelle la durée pendant laquelle la tâche t_i est bloquée *le temps de blocage* qui est notée B_i. Avec la spécification des paramètres temporelle des tâches et du protocole de gestion des ressources utilisé, le temps de blocage maximum d'une tâche peut être calculé. Lorsque le système nécessite des communications entre ces tâches s'a induit des contraintes de précédences. Ainsi, on dit que il y a une contrainte de précédence entre la tâche t_i et la tâche t_j ou t_i précède t_j, notée ($t_i \rightarrow t_j$), si t_j doit attendre la fin d'exécution de la tâche t_i pour commencer son exécution. Dans ce modèle, on considère que les tâches périodiques ont une forme atomique (forme normale) [8]. L'idée est que l'attente des messages par les tâches se fait en début, et l'émission des messages se fait en fin des tâches. On considère aussi que les contraintes des précédences entre les tâches sont de nature simple (c.à.d. $t_i \rightarrow t_j \Rightarrow P_i = P_j$).

Nous utilisons dans la suite le concept « *tâches synchrones* » dans le bute de signifier que les tâches sont sous contraintes de précédences et de partage des ressources critiques.

C. Variation de la vitesse du processeur

À l'heure actuelle, de nombreux processeurs permettent la variation de la tension d'alimentation et de la fréquence de fonctionnement. On assume que la vitesse peut être variée continuellement de S_{min} jusqu'à la vitesse maximale supporté [12]. On normalise les vitesses par la vitesse maximale pour

avoir un intervalle continu de vitesse [S_{min}, 1], où $S_{min} = f_{min} / f_{max}$ avec la fréquence minimale et maximale représenté par f_{min} et f_{max} respectivement.

III. ORDONNANCEMENT TEMPS REEL DES TACHES HYBRIDES, SYNCHRONES SOUS QUALITE DE SERVICE

Nous considérons dans cette section le problème d'ordonnancement des tâches hybrides (périodiques fermes et apériodique) synchrones (contraintes de précédences, de ressources). L'objectif est d'obtenir l'ordre d'exécution des tâches qui garanti le respect des contraintes de synchronisation et de qualité de service. Et aussi, d'avoir le meilleur temps de réponse des tâches apériodiques qui accèdent aussi aux ressources critiques en exclusion mutuelle.

A. Contraintes de précédences

Nous exploitons l'approche qui a été proposée dans [8] afin de prendre en compte les contraintes de précédence entre les tâches lors de l'utilisation de la politique EDF. Cette approche suggère la modification des paramètres dates de réveil et échéances afin que les contraintes de précédences soient implicitement respectées. Ces modifications sont opérées comme suit :

- Le calcul des dates de réveils des tâches s'effectue comme suit :

$$r_{i,0}^{*} = max\{r_{i,0}, max_{t_j \to t_i}\{r_{j,0}^{*} + C_j\}\} \quad (4)$$

A partir des tâches sans prédécesseurs et en descendant à chaque pas vers les tâches dont tous les prédécesseurs ont été traités.

- Quand aux échéances sont traités comme suit :

$$D_i^{*} = min\{D_i, min_{t_j \to t_i}\{D_j^{*} - C_j\}\} \quad (5)$$

A partir des tâches sans successeurs et en remontant à chaque pas vers les tâches dont tous les successeurs ont été traités.

Notons que ces modifications sont opérées en hors ligne. C'est-à-dire, avant l'ordonnancement effectif des tâches.

B. Contraintes de précédences et du partage des ressources

Nous avons choisi d'utiliser le protocole d'allocation de la pile SRP proposé dans [15], d'une part, pour gérer l'accès aux différentes ressources en exclusion mutuelle. Et d'autre part pour borner le temps de blocage des tâches en tenant en compte les contraintes de précédences. Ce protocole se base sur certains principes, lesquels sont résumés dans ce qui suit :

- En plus de sa priorité, chaque tâche se voit attribuer, de façon statique (hors ligne), un paramètre π entier appelé niveau de préemption. Les niveaux de préemption des tâches sont assignés de façon inversement proportionnelle à l'échéance.
- Chaque ressource se voit assigner dynamiquement une valeur plafond notée C_{Rk}, déterminée par la valeur maximale des niveaux de préemption des tâches actives ayant besoin de plus d'instances de la ressource R_k, c'est-à-dire, les taches dont la demande en unités de sémaphore ne peut pas être satisfaite.

$$C_{Rk} = max_i\{\pi_i / t_i \quad se\ bloque\ en\ R_k\} \quad (6)$$

- Le plafond système noté Π, est la valeur maximale des plafonds des ressources en cours d'utilisation.

Une tâche t_i, peut préempter une autre tâche t_j si les conditions suivantes sont vérifiées :

- Son échéance absolue d_i est inferieure à celle de t_j ;
- Le niveau de préemption de t_i est supérieur à celui de t_j, $\pi_i > \pi_j$
- Son niveau de préemption est supérieur au plafond système, $\pi_i > \Pi$.

Pour analyser l'ordonnancement d'une configuration de tâches hybrides synchrones, nous nous sommes basé sur la condition suffisante d'ordonnancement établit par Baker dans [15], que nous adaptons comme suit :

$$\forall i = 1,..., n;\ \sum_{j=1}^{i} m_j \cdot C_j / k_j \cdot D_j + m_i \cdot B_i / k_i \cdot D_i \le 1 \quad (7)$$

Où B_i représente la durée du blocage maximale de la tâche t_i. Cette durée est donnée par la formule suivante :

$$B_i = max_{j,k}\{\beta_{j,k} / \pi_i > \pi_j\ et\ C_{Rk} \ge t_i\ et\ t_j \notin \{succ\ de\ t_i\}\} \quad (8)$$

Nous considérons que le temps de blocage d'une tâche périodique t_i, est égale à la plus longue section critique des tâches périodiques, dont le niveau de préemption est inférieur à celui de t_i, et qui utilisent une ressource ayant la valeur plafond supérieure ou égale au niveau de préemption de t_i. En plus, les tâches de niveau de préemption inférieur considérées sont celles qui ne sont pas liées, directement ou indirectement, par des contraintes de précédences avec la tâche t_i.

C. Les tâches apériodiques

Lorsque le système est composé de tâches périodiques fermes et apériodiques qui partagent des ressources critiques, nous considérons les deux approches suivantes :

1) La première approche

Nous considérons que les tâches apériodiques sont gérées par une tâche serveur qui est périodique de priorité supérieure, dont la durée d'exécution est égale à la somme des durées d'exécution des tâches apériodiques. Pour analyser l'ordonnancement d'une configuration de tâches hybrides synchrones tout en garantissant une qualité de service (QoS), nous nous sommes basé sur la condition suffisante d'ordonnancement établit dans [15], que nous adaptons comme suit :

$$\forall i = 1,..., n+1;\ \sum_{j=1}^{i} m_j \cdot C_j / k_j \cdot D_j + m_i \cdot B_i / k_i \cdot D_i \le 1 \quad (9)$$

Où n+1 représente le nombre de tâches périodiques en plus de la tâche serveur. B_i représente la durée du blocage maximale de la tâche t_i. . Cette durée est donnée par la formule (8)

2) La deuxième approche

Notre deuxième approche d'ordonnancement des tâches apériodiques repose sur l'idée développée par Caccamo et al dans [9]. Ces derniers ont suggéré d'assigner à chaque requête apériodique une échéance et un niveau de préemption afin qu'elles puissent être ordonnancées par le protocole SRP avec l'utilisation du serveur à largeur de bande maximale noté TBS. L'attribution d'échéance s'effectue suivant la formule :

$$d_k = max(r_k, d_{k-1}) + C_{ak} / U_s \tag{10}$$

Où r_k est la date d'apparition de la k^{ème} tâche apériodique, d_k son échéance calculée, C_{ak} sa durée d'exécution, d_{k-1} l'échéance de l'apériodique précédent (d_0 = 0) et U_s est la charge du serveur utilisé. Cette charge admet pour valeur $U_s = 1 - U$. Cette attribution d'échéance respecte le fait que l'utilisation du processeur par les apériodiques ne dépasse jamais la valeur prédéfinie du serveur U_s. Quant au niveau de préemption, il est calculé comme suit :

$$\pi = U_s / C_{ak} \tag{11}$$

Pour l'analyse de l'ordonnancement des tâches hybrides synchrones tout en garantissant une qualité de service, nous adaptons la condition analytique d'ordonnancement qui a été proposé par Lipari et al dans [6]. Dans cette condition qui suit, nous considérons que les tâches sont en ordre croissant de leurs niveaux de préemption:

$$U + U_s \le 1 \tag{12}$$

$$\forall\, i, 1 \le i \le n,\ D_i \ge \sum_{j=1}^{i} \left\{ \left\lfloor \frac{D_i - D_j}{P_j} \right\rfloor + 1 \right\} \frac{m_j C_j}{k_j} + max\left\{ 0, \frac{m_i B_i}{k_i} - 1 \right\} + S(i) D_i U_s \tag{13}$$

Avec *S(i)* est la fonction de sélection qui égale à :

$$S(i) = \begin{cases} 0 & si\ \pi_i \ge \pi^* \\ 1 & si\ \pi_i < \pi^* \end{cases} \tag{14}$$

Où π* représente le niveau de préemption maximum des tâches apériodiques, π_i est le niveau de préemption de la tâche périodique t_i. D_i est l'échéance de la tâche t_i, D_j et P_j sont respectivement l'échéance et la période de la tâche périodique t_j ayant un niveau de préemption inférieur à celui de t_i. m_i est le nombre d'échéances que la tâche t_i doit respecter et k_i est le nombre d'activations consécutives. U est le facteur d'utilisation des tâches périodiques, U_s est la charge du serveur utilisé. Le terme B_i représente le temps de blocage maximum d'une tâche périodique t_i. Ce terme est calculé selon la formule (8).

IV. ADAPTATION DYNAMIQUE DE LA TENSION

Dans cette section nous calculons les facteurs d'utilisation du processeur lors de l'ordonnancement des tâches hybrides synchrones. Nous donnerons les deux algorithmes de calcul du facteur de vitesse du processeur suivant les deux approches présentées précédemment.

A. Algorithme I

Nous présentons dans ce paragraphe l'algorithme de calcul du facteur de vitesse du processeur noté S_f, lorsque la première approche est utilisée. Le facteur S_f est calculé de tel sorte que la condition analytique suivante soit vérifiée.

$$\forall i = 1,\ldots,n+1;\ \sum_{j=1}^{i} m_j \cdot C_j / k_j \cdot D_j \cdot S_f + m_i \cdot B_i / k_i \cdot D_i \cdot S_f \le 1 \tag{15}$$

Dans notre approche, une vitesse unique sera associée à l'ensemble des tâches et elle est déterminée avant l'exécution réelle des tâches. Ainsi, si le facteur de vitesse S_f est associé à la tâche t_i alors sa nouvelle durée d'exécution devienne $C_i * m_i / S_f * k_i$. L'algorithme qui calcul se facteur de vitesse est le suivant.

Algorithme I Calcul du facteur de vitesse S_f

1 $T = \{t_1, \ldots, t_n\}$ Etant donné un ensemble de n tâches périodiques fermes en ordre croissant de leurs échéances avec leur caractéristiques (C_i, D_i, B_i, m_i, k_i) ;
2 $X \leftarrow 0$, $Y \leftarrow 0$, $M \leftarrow 0$, $K \leftarrow 0$, $i \leftarrow 1$, $V \leftarrow 0$ {initialisation}
3 **while** ($i \le n$) **do**
4 $j \leftarrow 1$;
5 **while** ($j \le i$) **do**
6 M$\leftarrow m_j * C_j$
7 K$\leftarrow k_j * D_j$;
8 $X += M / K$
9 $j \leftarrow j+1$;
10 **end while**
11 M$\leftarrow m_i * B_i$
12 K$\leftarrow k_i * D_i$
13 $Y \leftarrow M / K$;
14 **if** ($X+Y > 1$) **then**
15 Exit;
16 **end if**
17 $V = Max(V, X+Y)$;
18 $X \leftarrow 0$;
19 $i \leftarrow i+1$;
20 **end while**
21 $S_f \leftarrow V$

B. Algorithme II

Shin et al ont mis en œuvre un algorithme dans [4] pour traiter le problème de la détermination du facteur vitesse du processeur, lors de l'ordonnancement des tâches périodiques et apériodiques avec le serveur TBS. Cet algorithme calcule deux facteurs de vitesse pour déterminer la vitesse d'exécution des tâches. Le premier concerne les tâches périodiques noté S_p, l'autre noté S_s est celui des tâches apériodiques. Cependant, les auteurs n'ont pas pris en compte les contraintes de ressources et de précédences. Pour déterminer les facteurs de vitesse du

processeur, nous utilisons le modèle présenté dans la deuxième approche. Nous formulons le problème comme suit :

Etant donné : U, U_s, w, ρ, $\{t_1,...,t_n\}$ ensemble des n tâches périodiques ordonnées avec leurs caractéristiques (C_i, P_i, B_i, D_i, m_i, k_i), π_i le niveau de préemption de la tâche t_i , π^* le niveau de préemption maximum des tâches apériodiques.

Trouver : S_s et S_p qui minimisent l'énergie E :

$$E = S_p^2 * w + S_s^2 * \rho \tag{16}$$

Telle que :

$$U_s / S_s + U / S_p = 1 \tag{17}$$

$$\forall i, 1 \le i \le n \quad D_i \ge \sum_{j=1}^{i} \left\{ \left\lfloor \frac{D_i - D_j}{P_j} \right\rfloor + 1 \right\} \frac{m_j C_j}{k_j S_p} +$$

$$max\left\{0, \frac{m_i B_i}{k_i S_p} - 1\right\} + S(i) D_i \frac{U_s}{S_s} \tag{18}$$

$$0 < S_p, S_s < 1 \tag{19}$$

Où : w est la charge moyenne des n tâches périodiques ($w = U / n$) et ρ est la charge moyenne des tâches apériodiques ($\rho = \lambda / \mu$), λ est le taux moyen d'arrivée et μ est le taux moyen de service. S_p et S_s sont les deux facteurs de vitesses.

Nous donnons la solution à ce problème par l'utilisation de la transformation de Lagrange comme suit :

$$S_p = U + U_s \sqrt[3]{\frac{\rho}{U_s * w}} \quad ; \quad S_s = U_s + U \sqrt[3]{\frac{U_s * w}{\rho}} \tag{20}$$

V. RESULTATS EXPERIMENTAUX

Dans cette section nous décrivons les simulations effectuées et les résultats expérimentaux obtenus dans le bute de d'évaluer les deux approches décrits précédemment et de comparer entre les deux algorithmes d'adaptation dynamique de la tension développés dans le cadre de notre étude.

A. Envirenement et contexte de simulation

Nous avons développé un simulateur avec le langage VC^{++}, dans lequel nous avons implémenté la politique d'ordonnancement EDF, le protocole de gestion des ressources critiques SRP et les algorithmes que nous avons développé. Pour réaliser les simulations, nous avons généré aléatoirement l'ensemble des tâches. Pour cela, nous avons générés selon la loi uniforme, les caractéristiques concernant les tâches périodiques. Le choix de la loi uniforme est motivé par le fait que les caractéristiques des tâches périodiques sont uniformément distribuées. Ainsi, les périodes (P_i) sont générées dans l'intervalle [40, 80], les durées d'exécution (C_i) dans l'intervalle [2, 8]. Les ressources (R_i) (de 0 à 2) et la position de la section critique est générée aléatoirement. La longueur de la section critique ($\beta_{i,k}$) de la tâche t_i qui accède au sémaphore S_k est générée dans l'intervalle [1, C_i]. La relation de précédence de t_i (de 0 à n-1) est générée aléatoirement. Les m_i et k_i de la contrainte (m_i, k_i)-firm sont générées aléatoirement. Les k_i sont uniformément distribués dans l'intervalle [4, 10] et m_i dans l'intervalle [2, k_i]. Dans le cas des simulations l'instance d'une tâche est rejetée par le système si elle est énumérée comme étant optionnelle par le modèle donné par l'équation 3.

Quant aux tâches apériodiques, leurs durées de service sont générées aléatoirement suivant la loi exponentiel de paramètres μ (= 0.25), et l'intervalle de temps entre deux arrivées est généré selon la loi de poisson de paramètre λ (= 0.01), car l'arrivés des tâches apériodiques suivent en général la loi de poisson. Tel que $1/\lambda$ est la durée moyenne des inter-arrivées, $1/\mu$ est la durée moyenne de service. Le temps de réponse moyen théorique des tâches apériodiques est calculé selon le modèle de Markov M/M/1 qui est de $(\mu - \lambda)^{-1} = 4.16$.

B. Résultats

Suivant les spécifications précédentes, nous avons généré 4 ensembles de 2 à 5 tâches périodiques avec une tâche apériodique et une autre de nature ferme dans chaque ensemble. Ces ensembles sont utilisés pour vérifier les deux approches et algorithmes proposés. Notons que dans le cas de la première approche nous avons généré en plus une tâche serveur ayant la plus haute priorité et dans le cas de la deuxième approche nous avons assigné au serveur TBS une charge 0.01. Les résultats des simulations sont donnés par les tableaux ci-dessous.

TABLE I. RÉSULTATS DE CALCUL DU FACTEUR DE VITESSE PAR L'ALGORITHME I

Ensembles de tâches	Algorithme I					
	Avant		Facteur de vitesse	Après		
	U	Tr_1	S_f	Tr_2	E. profit	Tqos
1	0.370	23.022	0.274	35.247	0,726	0.96
2	0.298	22.973	0.211	38.525	0,789	0.28
3	0.324	23.298	0.317	32.585	0,683	0.008
4	0.369	23.720	0.429	29.117	0,571	0.005

TABLE II. RESULTATS DE CALCUL DU FACTEUR DE VITESSE PAR L'ALGORITHME I (U_s= 0.01)

Ensembles de tâches	Algorithme II							
	Avant			Facteur de vitesse		Après		
	U_s	U	Tr_1	S_s	S_p	Tr_2	E. profit	Tqos
1	0.01	0.10	4.36	0.04	0.138	264.29	0.303	0.45
2	0.01	0.19	5.16	0.08	0.220	258.10	0.225	0.41
3	0.01	0.21	5.35	0.09	0.245	377.60	0.110	0.03
4	0.01	0.29	6.31	0.13	0.323	340.20	0.051	0.01

Nous remarquons que l'algorithme I (première approche) donne le meilleur temps de réponse moyen noté T_r, le gain en énergie comparativement à l'algorithme II (second approche). Quand au taux de qualité de service, nous remarquons que la première approche garantie un taux élevé lors de l'ordonnancement du premier ensemble de tâches. Cependant, ce taux diminue à mesure que le nombre de tâche augmente.

Cette diminution est remarquée aussi avec la deuxième approche mais avec un degré moins.

Lorsque nous affectons une charge au serveur TBS (U_s=0.3) comme montré par le tableau III ci-dessous, nous remarquons la deuxième approche offre le meilleur temps de réponse moyen de tâche apériodique mais avec un gain en énergie toujours plus important dans le cas de la première approche. Nous remarquons aussi que le taux de la qualité de service ne change pas avec le changement de la charge attribuée au serveur TBS, et ceci est dû au fait que ce taux dépend uniquement des paramètres m_i et k_i des tâches selon le modèle donnée par l'équation 3.

TABLE III. RESULTATS DE CALCUL DU FACTEUR DE VITESSE PAR L'ALGORITHME II (U_S= 0.3)

Ensembles de tâches	Algorithme II							
	Avant			*Facteur de vitesse*		*Après*		
	U_s	U	Tr_1	S_s	S_p	Tr_2	*E. profit*	*Tqos*
1	0,3	0,1	4	0,4	0,43	10,097	0,596	0.45
2	0,3	0,19	4,07	0,52	0,46	7,738	0,508	0.41
3	0,3	0,22	4,07	0,56	0,47	7,169	0,481	0.03
4	0,3	0,3	4,08	0,69	0,53	5,857	0,401	0.01

VI. CONCLUSION

Dans cet article, nous avons visé l'intégration des contraintes de qualité de services dans l'ordonnancement des tâches périodiques et apériodiques sous contraintes de synchronisation. Nous avons proposé deux algorithmes d'adaptation de la tension du processeur dans le même contexte. Nous avons montré que les deux approches proposées donnent des résultats intéressant en termes de simulation.

VII. REFERENCE

[1] A. Abbas & H. Hentous "Ordonnancement temps réel des tâches hybrides synchrones sous contrainte d'énergie ", in proc. International Symposium on Operational Research, ISOR'08, pp. 607-618, Alger, Algeria : November 2008.

[2] A. Abbas, H.Hentous & T.Kenaza "Scheduling real-time of the synchronous hybrid tasks under energy constraint" in proc. Third International Conference on Sensor Technologies and Applications, SENSORCOMM'09, Athens, Glyfada, Greece, p. 240-245, June 2009.

[3] C.L. Liu and J.W. Layland. "Scheduling algorithms for multiprogramming in real-time environment". Journal of the ACM, 1973, pp. 20(1): 46–61.

[4] D. Shin and J. Kim, "Dynamic Voltage Scaling of Periodic and Aperiodic Tasks in Priority-Driven Systems," in Proc. ASPDAC'03, Jan. 2004, pp. 653-658.

[5] F. Yao, A. J. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy", in IEEE Symposium on Foundations of Computer Science, 1995, pp. 374-382.

[6] G. Lipari and G. Buttazzo, "Schedulability analysis of periodic and aperiodic tasks with resource constraints", Journal of Systems Architecture, 2000, Vol. 46, No. 4, pp. 327-338.

[7] H. Aydin, R. Melhem, D. Mossé and P. Mejia-Alvarez, "Determining optimal processor speeds for periodic real-time tasks with different power characteristics" Euromicro Conference on Real-Time Systems, 2001, pp. 225-232.

[8] J. Blazewicz "Scheduling dependent tasks with different arrival times to meet deadlines" in E. Gelembre, H. Beiher Modelling and performance evaluation of computer systeme Nort-holland, amsterdem, 1976.

[9] M. Caccamo, G. Lipari, and G. Buttazzo, "Sharing resources among periodic and aperiodic tasks with dynamic deadlines", Proceedings of the IEEE Real-Time Systems Symposium, Phoenix, Arizona, 1999, pp. 284-293.

[10] M. Hamdaoui, P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines", IEEE Transactions on Computers 44, December 1995, p. 1443–1451.

[11] M. Spuri and G.C. Buttazzo, "Efficient Aperiodic Service under Earliest Deadline Scheduling", Proc.of the IEEE Real-Time Systems Symposium, San Juan, Portorico, December 1994.

[12] Nasro min-allah, Yong-ji wang, jian-sheng xing, wasif nisar and asad-raza kazmi " Towards Dynamic Voltage Scaling in Real-Time Systems – A Survey", IJCSES International Journal of Computer Sciences and Engineering Systems, Vol.1, No.2, CSES International ISSN 0973-4406, April 2007.

[13] P. Ramanathan. Overload management in real-time control applications using (m,k)-firm guarantee. IEEE Trans. on Paral. and Dist. Sys., 10(6):549–559, Jun 1999.

[14] R. Jejurikar and R. Gupta, "Energy Aware Task Scheduling with Task Synchronization for Embedded Real Time Systems ", Proc. Int'l Conf. Compilers, Architecture and Synthesis for Embedded Systems, 2002.

[15] T.P. Baker. "Stack-based scheduling of real-time processes". The Journal of Real-Time Systems, 1991, pp. 3 :67–99.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.